Homework 10

The purpose of this homework is to learn about methods selecting models and to practice data visualization. Please fill in the appropriate code and write answers to all questions in the answer sections, then submit a compiled pdf with your answers through gradescope by 11:59pm on Sunday November 24th.

As always, if you need help with any of the homework assignments, please attend the TA office hours which are listed on Canvas and/or ask questions on Piazza. Also, if you have completed the homework, please help others out by answering questions on Piazza.

Part 1: Model selection

In class 20 we fit polynomial models of up to degree 5 for predicting the price of a used Toyota Corolla from the number of miles driven. We showed that the R^2 value for this fit always increases (or stays the same) as more variables (or a higher degrees polynomials) are added to the model. Thus if one was to judge how "good" a model was based on the R^2 statistic value, one would always choose a very complex model that has likely has a lot of variables.

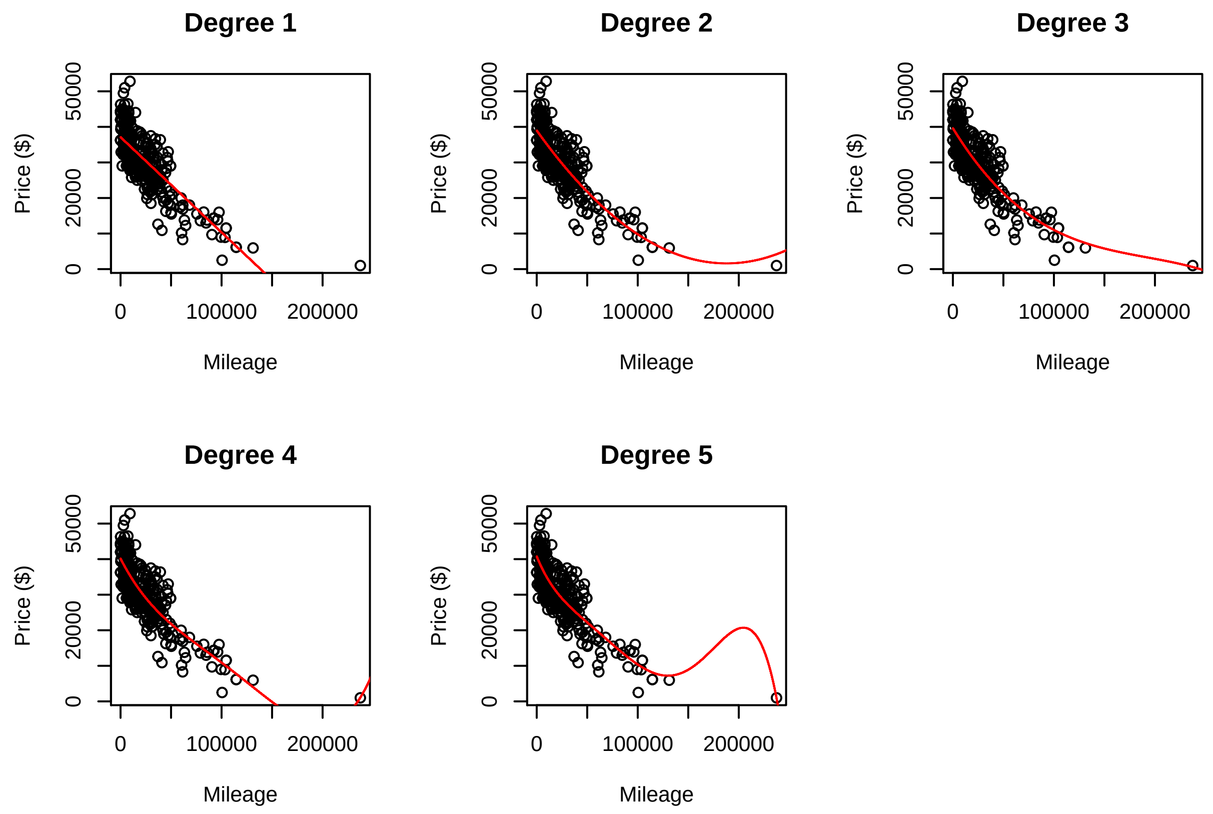
As we also discussed in class, there are other statistics and methods that can potentially give better measures of how well a model fits the data. In the set of exercises below, you will empirically evaluate these different methods for selecting models to see how well they work.

Part 1.1 (3 points):

The code below loads the Edmunds car transaction data and creates a data frame that has data from the used BWM model '3 Series'. It also plots the model fits for predicting price as a function of miles driven for models up to degree 5.

Based on looking at the model fits, which degree polynomial do you think is the best fit to the data?

```r
load('car_transactions.rda')
car_model_name <-  "3 Series"   #  "MAZDA3"
used_cars <- select(car_transactions, price_bought,mileage_bought,
                    model_bought, make_bought, new_or_used_bought) %>%
  filter(model_bought == car_model_name, new_or_used_bought == "U") %>%
  na.omit(.)
par(mfrow = c(2, 3))
x_vals_df <- data.frame(mileage_bought = 0:300000)
for (i in 1:5){

  curr_model <- lm(price_bought ~ poly(mileage_bought, degree = i), data = used_cars)

  model_summary <- summary(curr_model)

  y_vals_predicted <- predict(curr_model, newdata = x_vals_df)
  plot(price_bought ~ mileage_bought, data = used_cars, xlab = "Mileage",
       ylab = "Price ($)", main = paste("Degree", i))

  points(x_vals_df$mileage_bought, y_vals_predicted, type = "l", col = "red")


}
```



Answers Just looking at the model fits, it seems that the Degree 3 polynomial is the best fit to the data. I believe the degree 3 polynomial fits the points the most accurately since it is continually decreasing (unlike degree 2, 4, and 5 polynomials, although the 2nd degree does seem to be a good fit also) and accounts for the curvature when the mileage increases to very high.

Part 1.2 (10 points): Now let's try calculating different measure of model fit for polynomials from degree 1 to degree 5. Use a for loop to that loops over models of degree 1 to degree 5 and saves the following statistics:

1) R^2: save to a vector called `all_r_squared`
2) R^2_{adj}: save to a vector called `all_r_squared`
3) AIC: save to a vector called `all_aic`
4) BIC: save to a vector called `all_bic`

Then use the which.max() function and which.min() function to determine which model each of these statistics would select. Fill in the table below by placing an x in the appropriate column for the model that each statistic would select (you will fill in the last line in part 1.3). Also comment on which statistics you think are leading to the best model choice.

```r
all_r_squared <- NULL
all_adj_r_squared <- NULL
all_aic <- NULL
all_bic <- NULL
# create a for loop to extract the relevant statistics
# print the degree selected by each model selection method
#using the which.max() or which.min() functions

for (i in 1:5){

  curr_model <- lm(price_bought ~ poly(mileage_bought, degree = i), data = used_cars)

  model_summary <- summary(curr_model)

  all_r_squared[i] <- model_summary$r.squared
  all_adj_r_squared[i] <- model_summary$adj.r.squared
  all_aic[i] <- AIC(curr_model)
  all_bic[i] <- BIC(curr_model)

}


which.max(all_r_squared)
```

```
## [1] 5
```

```r
which.max(all_adj_r_squared)
```

```
## [1] 5
```

```r
which.min(all_aic)
```

```
## [1] 3
```

```r
which.min(all_bic)
```

```
## [1] 2
```

Answer

	1	2	3	4	5
R^2					x
R^2_{adj}					x
AIC			x		
BIC		x			
cross-val		x			

Answer

I think the AIC and BIC models are leading to the best model choice since they are penalizing models with a greater number of predictors (as we can see from the graphs, the degree 5 polynomial is not a very good predictor as the R^2 values may suggest). I think in this case AIC is a bit better than BIC because even thought BIC allows us to choose a model with less predictors, it seems that the third degree polynomial is a bit of a better fit than the second.

Part 1.3 (10 points):

As we also discussed in class, we can use cross-validation to assess model fit, but building a model on a *training set* of data and then evaluating the accuracy of the predictions on a separate *test set* of data. When evaluating the model, we can use the *mean squared prediction error* (MSPE) as a measure of how accurately the model is making predictions on new data. This measure is defined as:

$MSPE = \frac{1}{m}\Sigma_{i=1}^{m}(y_i - \hat{y}_i)^2$ where the y_i come from the m points in the *test set*.

For more information on this measure, see wikipedia.

The code below splits the data in half and creating a *training set* that we will use to learn the estimated coefficients $\hat{\beta}$'s and also creating a *test set* with the other half of the data that we can evaluate how accueratedly the model can make predictions. Use a for loop to create models of degree 1 to 5 using the training data, have the models predictions on the test set, and then calculate the MSPE based on their predictions. Add to the table above when model has the minimum MPSE by putting an x in the appropriate column and print out the result below to show your work.

```r
# create the training set and the test set
total_num_points <- dim(used_cars)[1]
num_training_points <- floor(total_num_points/2)
training_data <- used_cars[1:num_training_points, ]
test_data <- used_cars[(num_training_points + 1):total_num_points, ]
# run a for loop to calculate the MSPE for models of degree 1 to 5
MSPE_values <- c()
for(i in 1:5){
  fit_cv <- lm(price_bought ~ poly(mileage_bought, degree = i), data = training_data)
  test_predictions_1 <- predict(fit_cv, newdata = test_data)
  MSPE_values[i] <- mean((test_data$price_bought - test_predictions_1)^2)
}
# then find the model with the minimal MSPE
which.min(MSPE_values)
```

```
## [1] 2
```

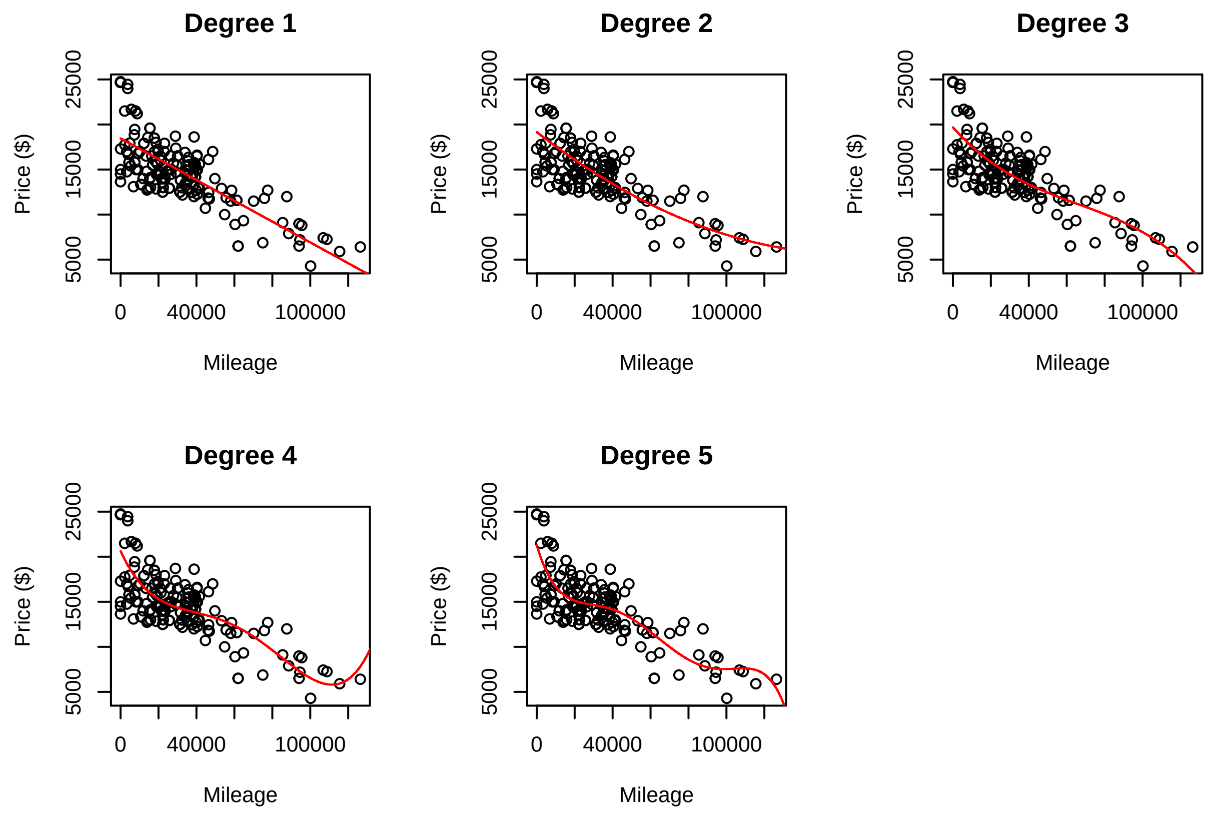
Part 1.4 (10 points):

Now rerun parts 1.1 to 1.3 but using only Mazda 3's instead of BMW 3 series (e.g., filter for "MAZDA3"). If you have written your code in a flexible way you should really only have to change the line "3 Series" to "Mazda 3" in part 1.1 and the rest of the code should run. Then fill out the table below for Mazda 3's, and answer all the following questions for the Mazda 3 data:

a) From looking at the models fits, which degree model seems to fit the data best
b) which statistics you think are leading to the best model choice

c) overall (for both car brands) which model selection method do you think is working best (and why).

```
#Part 1.1
car_model_name <-  "MAZDA3"
used_cars <- select(car_transactions, price_bought,mileage_bought, model_bought, make_bought, new_or_use
  filter(model_bought == car_model_name, new_or_used_bought == "U") %>%
  na.omit(.)
par(mfrow = c(2, 3))
x_vals_df <- data.frame(mileage_bought = 0:300000)
for (i in 1:5){

  curr_model <- lm(price_bought ~ poly(mileage_bought, degree = i), data = used_cars)

  model_summary <- summary(curr_model)

  y_vals_predicted <- predict(curr_model, newdata = x_vals_df)

  plot(price_bought ~ mileage_bought, data = used_cars, xlab = "Mileage",
       ylab = "Price ($)", main = paste("Degree", i))

  points(x_vals_df$mileage_bought, y_vals_predicted, type = "l", col = "red")
}
```



```
#Part 1.2
all_r_squared <- NULL
all_adj_r_squared <- NULL
```

```r
all_aic <- NULL
all_bic <- NULL
# create a for loop to extract the relevant statistics
# print the degree selected by each model selection method
#using the which.max() or which.min() functions

for (i in 1:5){

  curr_model <- lm(price_bought ~ poly(mileage_bought, degree = i), data = used_cars)

  model_summary <- summary(curr_model)

  all_r_squared[i] <- model_summary$r.squared
  all_adj_r_squared[i] <- model_summary$adj.r.squared
  all_aic[i] <- AIC(curr_model)
  all_bic[i] <- BIC(curr_model)

}


which.max(all_r_squared)
```

```
## [1] 5
```

```r
which.max(all_adj_r_squared)
```

```
## [1] 5
```

```r
which.min(all_aic)
```

```
## [1] 5
```

```r
which.min(all_bic)
```

```
## [1] 5
```

```r
#Part 1.3
# create the training set and the test set
total_num_points <- dim(used_cars)[1]
num_training_points <- floor(total_num_points/2)
training_data <- used_cars[1:num_training_points, ]
test_data <- used_cars[(num_training_points + 1):total_num_points, ]
# run a for loop to calculate the MSPE for models of degree 1 to 5
MSPE_values <- c()
for(i in 1:5){
  fit_cv <- lm(price_bought ~ poly(mileage_bought, degree = i), data = training_data)
  test_predictions_1 <- predict(fit_cv, newdata = test_data)
  MSPE_values[i] <- mean((test_data$price_bought - test_predictions_1)^2)
}
# then find the model with the minimal MSPE
which.min(MSPE_values)
```

```
## [1] 2
```

Answer

	1	2	3	4	5
R^2					X
R^2_{adj}					X
AIC					X
BIC					X
cross-val		X			

a) From looking at the plots it seems that the degree 2 polynomial since it most closely resembles how we would expect car prices to decline.

b) All of the statistics except the Mean Squared Prediction Error point towards the same model (the 5th degree polynomial), which doesn't seem to be the best model choice. Therefore, MSPE seems to be the best statistic for choosing a model. Among the first four statistics we considered at first, I think AIC and BIC would tend to lead to the best model (although they did not) since they are penalizing for the number of predictors we have, whereas R^2 will always increase. Adjusted R^2 also accounts for the number of predictors but the values usually end up very similar to R^2. If we are looking for a smaller model, BIC is more effective than AIC.

c) It seems that cross-validation worked best across both models of cars since it was the only method which chose correctly for the Mazda 3's, and chose one of the two possible correct fits for BMWs. This makes sense as cross validation is testing whether the model actually works on the data, whereas other methods simply use all of the data to make a predictive model.

Bonus question (0 points):

Run three-fold cross-validation where split the data into 3 parts and you:

a) learn the model parameter estimates on two of the data splits (2/3rds of the data)
b) make predictions on the last data split (1/3 of the data)
c) repeat steps *a* and *b* leaving a different test set out each time and training on the other 2 splits
d) average the MSPE results over the 3 cross-validation splits.
e) repeat this procedure for models of degree 1-5 (i.e., by having nested for loops)

Does the three-fold cross-validation results lead to the same conclusions as using only 1 training and test split?

```r
# create the training set and the test set
total_num_points <- dim(used_cars)[1]
training_points_1 <- floor(total_num_points/3)
training_points_2 <- floor(2*total_num_points/3)
training_data_1 <- used_cars[1:training_points_1, ]
training_data_2 <- used_cars[(training_points_1+1):training_points_2, ]
training_data_3 <- used_cars[(training_points_2+1):total_num_points, ]

#First 2
# run a for loop to calculate the MSPE for models of degree 1 to 5
```

```r
MSPE_values <- c()
training_1 <- rbind(training_data_1, training_data_2)
training_2 <- rbind(training_data_1, training_data_3)
training_3 <- rbind(training_data_2, training_data_3)


for(i in 1:5){
  fit_cv_1 <- lm(price_bought ~ poly(mileage_bought, degree = i), data = training_1)
  test_predictions_1 <- predict(fit_cv_1, newdata = training_data_3)
  MSPE_val_1 <- mean((training_data_3$price_bought - test_predictions_1)^2)

  fit_cv_2 <- lm(price_bought ~ poly(mileage_bought, degree = i), data = training_2)
  test_predictions_2 <- predict(fit_cv_2, newdata = training_data_2)
  MSPE_val_2 <- mean((training_data_2$price_bought - test_predictions_2)^2)

  fit_cv_3 <- lm(price_bought ~ poly(mileage_bought, degree = i), data = training_3)
  test_predictions_3 <- predict(fit_cv_3, newdata = training_data_1)
  MSPE_val_3 <- mean((training_data_1$price_bought - test_predictions_3)^2)

  MSPE_values[i] <- mean(MSPE_val_1, MSPE_val_2, MSPE_val_3)
}
# then find the model with the minimal MSPE
which.min(MSPE_values)
```

```
## [1] 2
```

Answer

Training on two thirds of the data set still gives us the second degree polynomial as the best model.

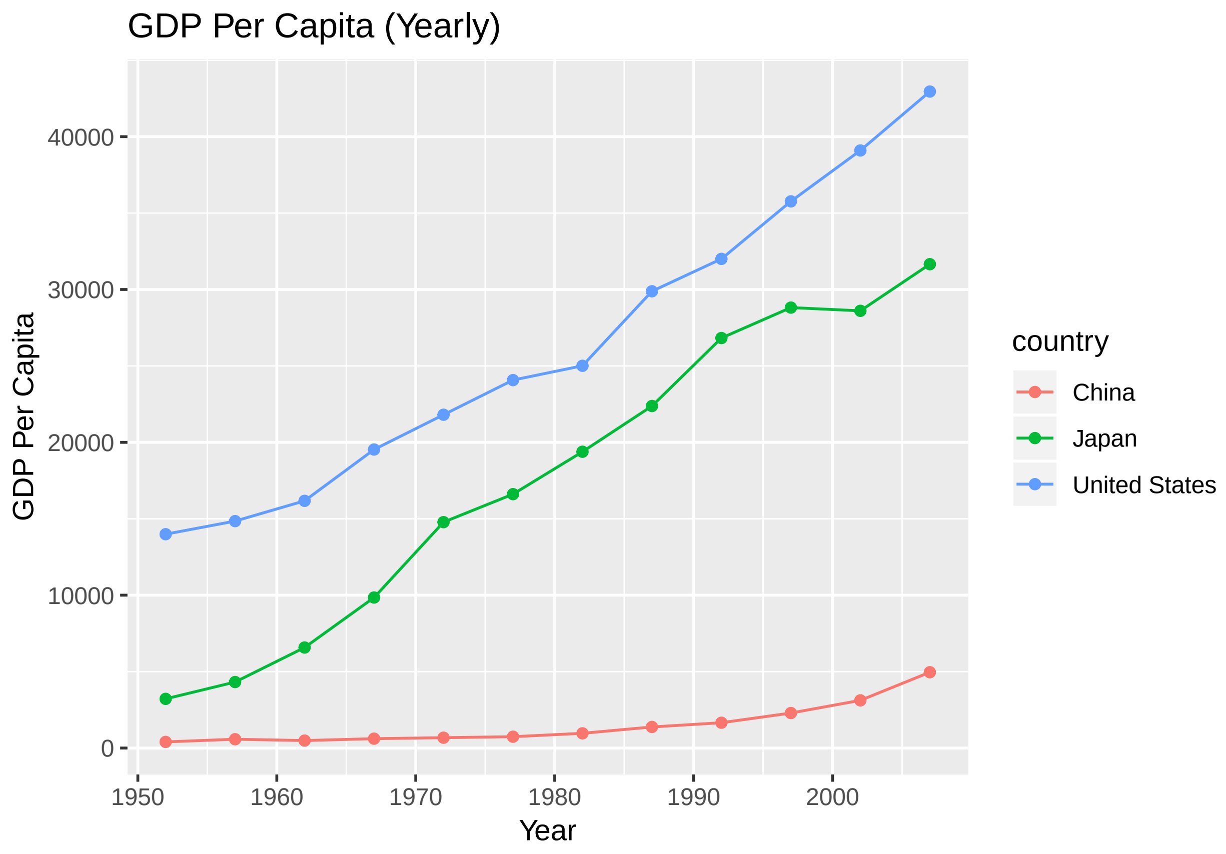
Part 2: Data visualization

In the next set of exercises you will use ggplot2 to compare different visualizations and see which gives the clearest insights. A useful resource for ggplot and other tidyverse code is the book R for Data Science.

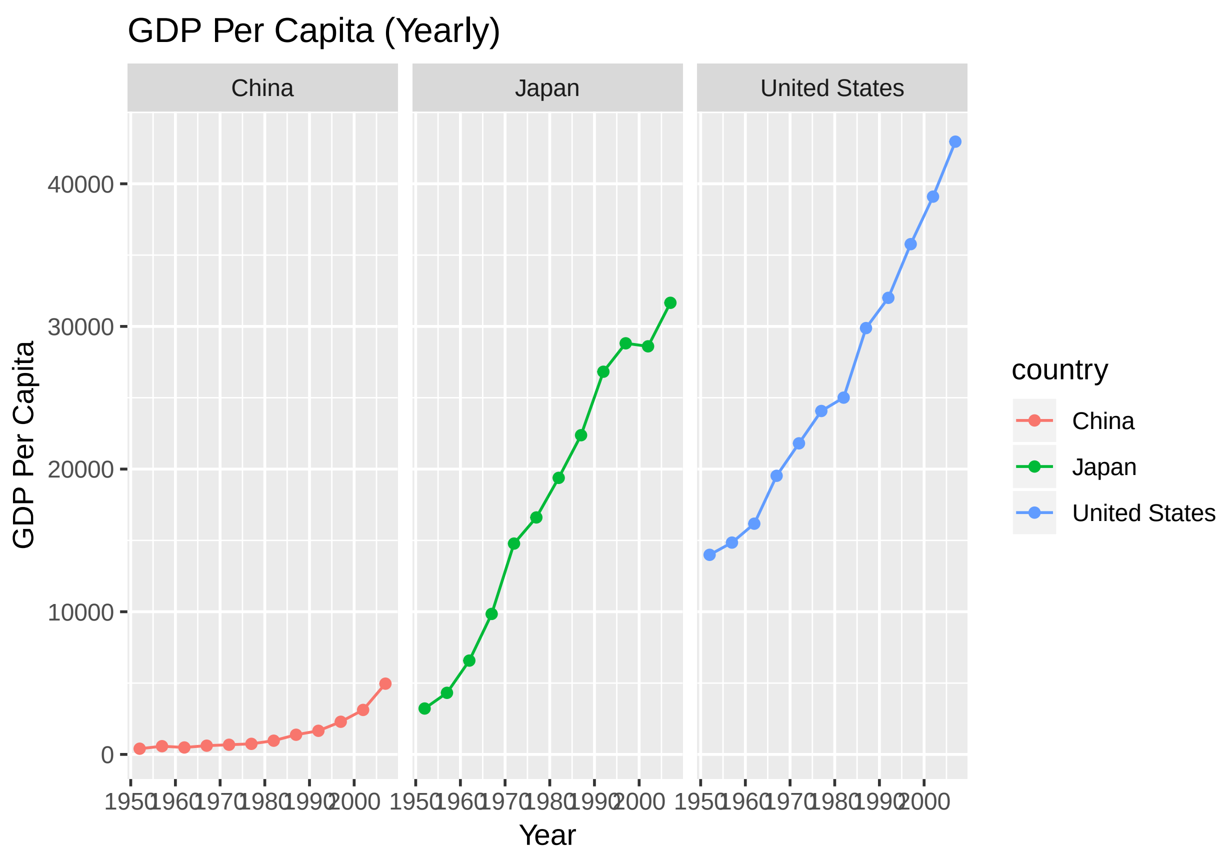
Part 2.1 (10 points): Let's start by comparing some visualizations on the gapminder data which contains information about different countries in the world over time. Use ggplot and the gapminder data to compare the GDP per capita of Japan, the United States and China. Plot a line graph of GDP per capita as a function of the year, with each country in a different color. Also, create a plot that compares these countries GDP per capita as a funciton of the year using facets, where the data from each country is on a separate subplot. As always, make sure to label your axes in this plot and in all other plots in this worksheet. Do you think one type of plots is better than another in comparing these countries? Explain why. (Hint for completing this exercise: first use the dplyr filter() function to get the subset of data you need, then plot it).

```r
# filter the data to get only data from Japan the United States and China
gapminder_filtered <- gapminder %>% filter(country == "Japan" |
                                             country == "United States" |
                                             country == "China")
```

```r
# create a line plot showing the three countries on the same plot
ggplot(gapminder_filtered, aes(x = year, y = gdpPercap, col = country)) +
  geom_line() + geom_point() + labs(fill = "Country", x = "Year",
  y = "GDP Per Capita", title = "GDP Per Capita (Yearly)")
```



```r
# use facets to put the three countries on side-by-side plots
ggplot(gapminder_filtered, aes(x = year, y = gdpPercap, col = country)) +
  geom_line() + geom_point() + labs(x = "Year",
  y = "GDP Per Capita", title = "GDP Per Capita (Yearly)") +
  facet_wrap(~country)
```



Answers: [Explain whether you think of of these plots is more informative than the other].

I think the second plot with each country on its own subplot is more informative. Since the scales for each plot are the same we can still see how each country's GDP per capita varies in magnitude, as with the first plot. However, the side-by-side plots give us a better sense for how rapidly the GDP of Japan and the United States grew as compared to China, and also show us that China's GDP has begun a steep rise (while still being far below either of the other two countries). Overall, I feel that the two plots are still pretty similar and show the same information.

Part 2.2 (10 points): DataExpo is a Statistics event at the Joint Statistical Meetings where different researchers compare data analysis methods applied to a common data set. In 2018, the data set consisted of weather predictions made between 2014 and 2017. In this exercise, let's look at the data from this event and try to visualize the prediction accuracies for predictions made 0 to 6 days in the future.

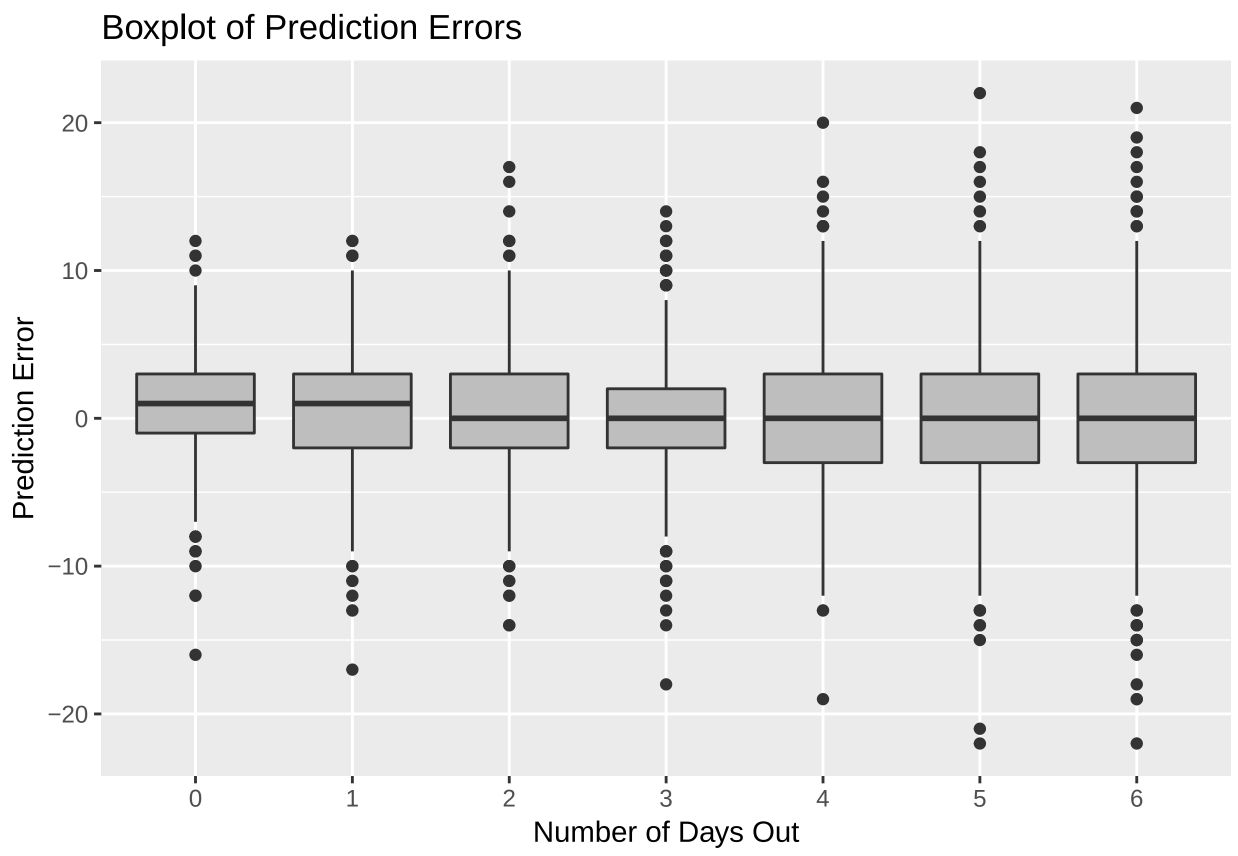
The code below loads a data frame called `forecast_ne_joined` that has the prediction errors for the maximum temperature for the 9 cities in New England, along with several other variables. First, create a new data frame called `new_haven_preds` that has only the predictions from New Haven, and has only the variables cityID, city, num_days_out_prediction_made and max_temp_prediction_error. Also, type convert the variable `num_days_out_prediction_made` to a factor using the mutate() and as.factor() functions. Then use ggplot to create plots that compare the prediction accuracy as a function of the number of days in advance that a prediction was made using the following geoms:

1) Create a boxplot using geom_boxplot()
2) Create a violin using geom_violin()
3) Create a joy plot using geom_density_ridges()

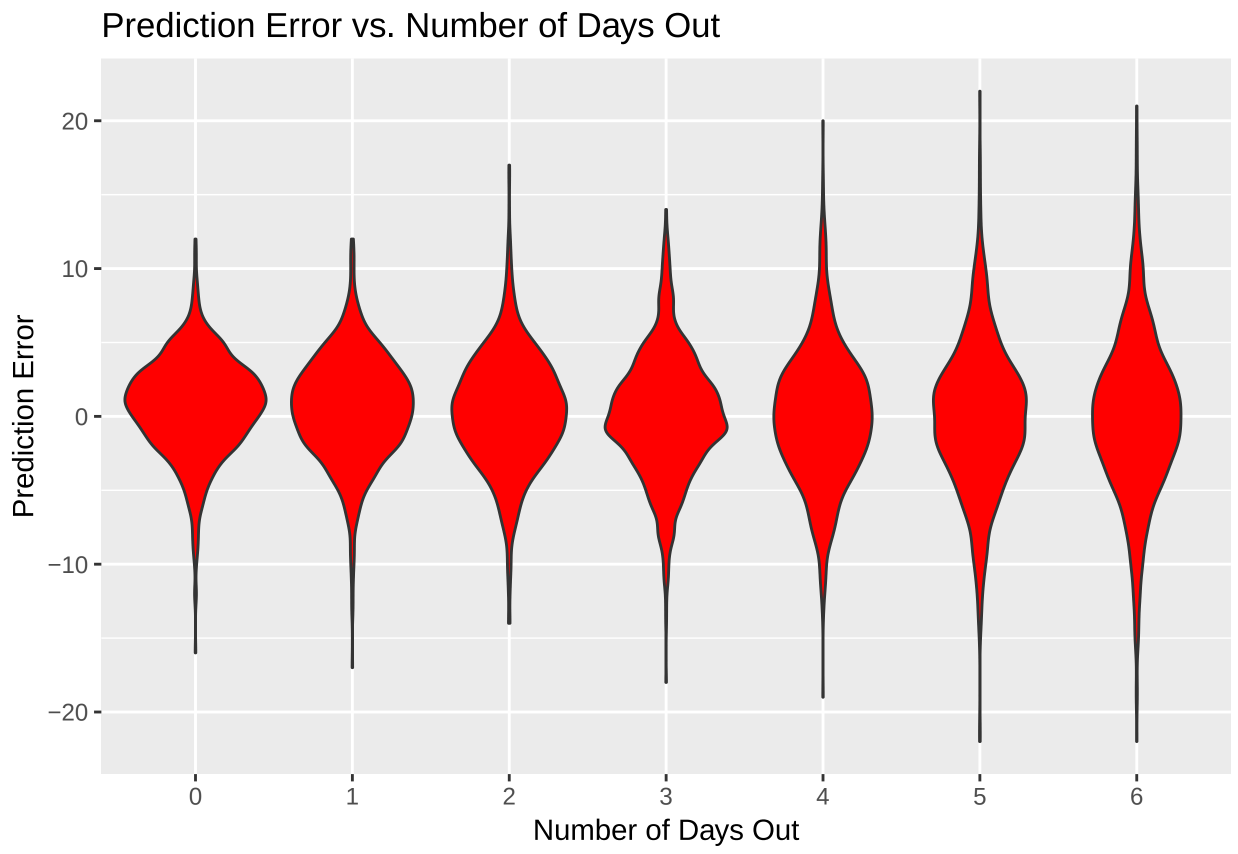
Note that the geom geom_density_ridge() comes from the ggridges packages that was loaded at the top of the worksheet, and that the x and y aesthetic mapping is in the opposite order as the mapping used for the geom_boxplot() and geom_violin() geoms.

After you created these plots, briefly discuss which plot you believe which most clearly shows how the prediction accuracy decreases as a function of days in the future. Also, don't forget to label your axes using the xlab() and ylab() functions.

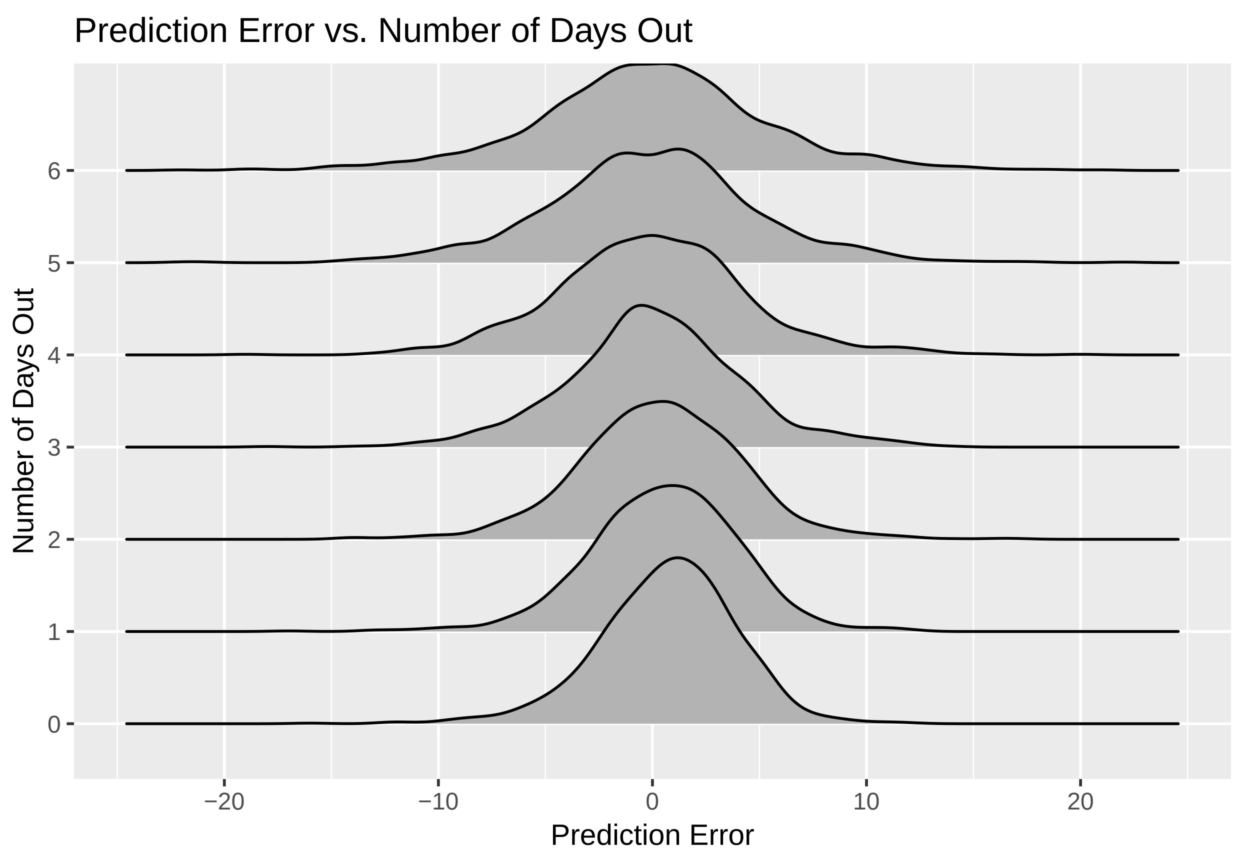
```r
# load the data that has the weather prediction errors
forecast <- read.csv('forecast_ne_joined.csv')
# filter and select the data to get data from only New Haven for the 4 variables of interest
forecast_filtered <- forecast %>% filter(city == "New Haven") %>%
  select(cityID, city, num_days_out_prediction_made, max_temp_prediction_error)
# and convert num_days_out_prediction_made to a factor using the mutate() function
forecast_filtered <- forecast_filtered %>% mutate(num_days_out_prediction_made =
                                                       as.factor(num_days_out_prediction_made))
# compare the predictions made 0 to 6 days out using box plots
ggplot(forecast_filtered, aes(x = num_days_out_prediction_made,
                              y = max_temp_prediction_error)) +
                              geom_boxplot(fill = "gray") +
            xlab("Number of Days Out") + ylab("Prediction Error") +
            ggtitle("Boxplot of Prediction Errors")
```



```r
# compare the predictions made 0 to 6 days out using violin plots
ggplot(forecast_filtered, aes(x = num_days_out_prediction_made,
                              y = max_temp_prediction_error))+
  geom_violin(fill = "red") + labs(x = "Number of Days Out",
                                   y = "Prediction Error",
                                   title = "Prediction Error vs. Number of Days Out")
```



```
# compare the predictions made 0 to 6 days out using joy plots
ggplot(forecast_filtered, aes(y = num_days_out_prediction_made,
                              x = max_temp_prediction_error ))+
  geom_density_ridges() + labs(y = "Number of Days Out",
                               x = "Prediction Error",
                               title = "Prediction Error vs. Number of Days Out")
```



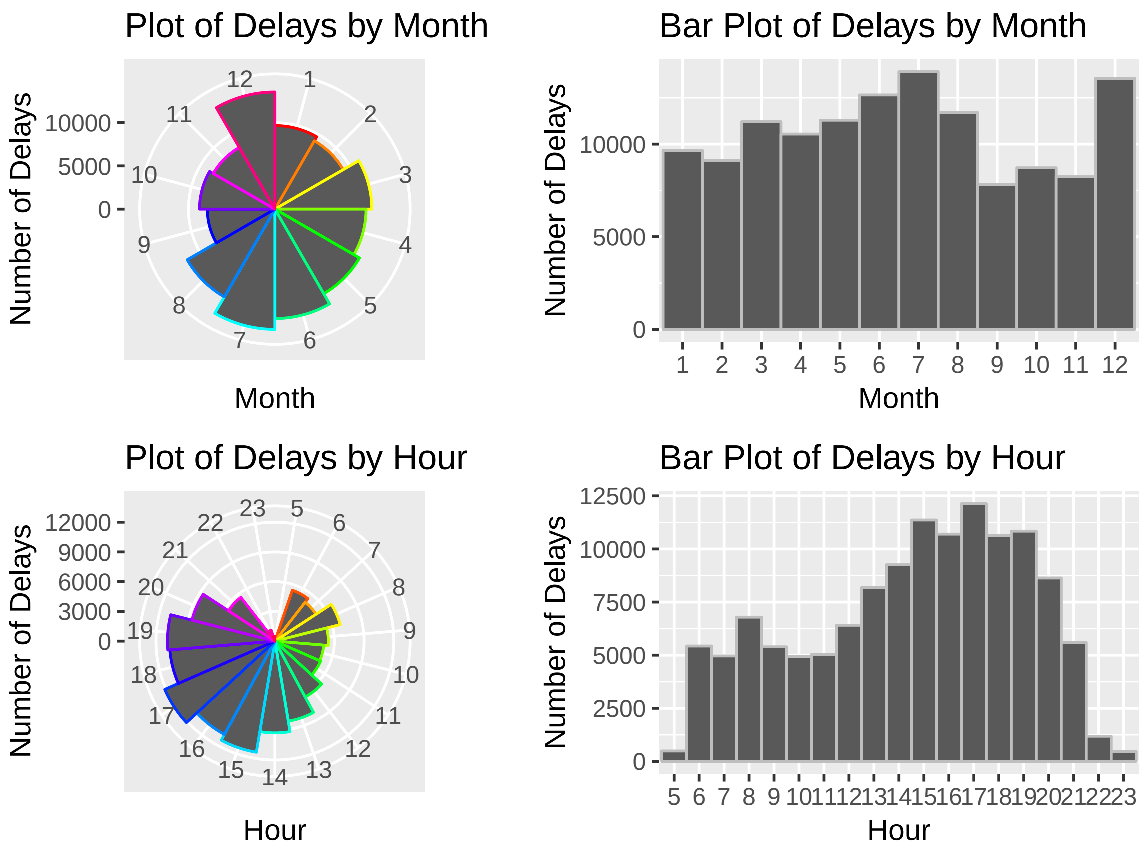
Answers:

I think the joy plot shows most clearly how the prediction accuracy decreases as a function of days in the future. From this plot we can see that the prediction error is clustered at a higher density around 0 when the Number of Days Out is lower, and the peak becomes shorter and wider as the number of days increases. This visual representation is very clear since we can see the distributions increasing progressively in width. I think this progression can also be seen with the violin and box plots, but is not as clear because we don't have a way of determining whether the points outside the middle 50 percent actually make the entire prediction error increase.

Part 2.3 (10 points): Create an *interesting* plot using one of the data sets we have discussed in class or another data set you can find. Try exploring other features of ggplot we have not discussed in class using the ggplot cheat sheet. See if you can find something interesting in the data and explain why you find it interesting.

```
require("nycflights13")
data(flights)
data(airlines)

#As in Pset 9, I took the flights data and filtered out delayed flights,
#then grouped by month and hour.
delayed_flights <- flights %>% filter(dep_delay > 0)

month_delay <- flights  %>%  group_by(month) %>%
   summarize(mean_arr_delay = mean(arr_delay, na.rm = TRUE))%>%
```

```r
  arrange(desc(mean_arr_delay))

hour_delay <- flights  %>%  group_by(hour) %>%
   summarize(mean_arr_delay = mean(arr_delay, na.rm = TRUE)) %>%
  arrange(desc(mean_arr_delay))

#Coxcomb Chart of each month and the number of delays
require(gridExtra)
plot1 <- ggplot(delayed_flights, aes(x = factor(month))) +
  geom_bar(width = 1, colour = rainbow(n = 12)) + coord_polar() +
  xlab("Month") + ylab("Number of Delays") + ggtitle("Plot of Delays by Month")
plot2 <- ggplot(delayed_flights, aes(x = factor(month))) +
  geom_bar(width = 1, colour = "gray") +
  xlab("Month") + ylab("Number of Delays") + ggtitle("Bar Plot of Delays by Month")

#Coxcomb chart of each hour and the number of delays
plot3 <- ggplot(delayed_flights, aes(x = factor(hour))) +
  geom_bar(width = 1, colour = rainbow(19)) + coord_polar()+
  xlab("Hour") + ylab("Number of Delays") + ggtitle("Plot of Delays by Hour")

plot4 <- ggplot(delayed_flights, aes(x = factor(hour))) +
  geom_bar(width = 1, colour = "gray") +
  xlab("Hour") + ylab("Number of Delays") + ggtitle("Bar Plot of Delays by Hour")

grid.arrange(plot1, plot2, plot3, plot4, ncol = 2)
```



Answers:

From the last pset we analyzed the flights dataset and found that for months, July had the longest average delay, and in hours, 9pm had the longest delays. However, using ggplot I created two Coxcomb charts showing the nubmer of total delays for the dataset grouped by month and by hour. After visualizing we can see that the highest *number* of delays occurred during the month of December and at the time of 5pm. I think this information is probably better presented in a bar plot which is why I included the barplots side by side with the Coxcomb plot. The Coxcomb plot is probably better for presentation if we are aiming to make more impact on a less knowledgeable audience. Another reservation I have with this visualization is that the chart only shows the number of delays and not the proportion. These numbers then absolutely make sense because more people travel during the summer and December (vacation months), and in the evenings, so we would expect more problems and delays at those times.

Reflection (3 points)

Please reflect on how the homework went by going to Canvas, going to the Quizzes link, and clicking on Reflection on homework 10